FOCUS: The Interactive Table for Product Comparison and Selection

Michael Spenke, Christian Beilken, Thomas Berlage

GMD — German National Research Center for Information Technology
FIT.MMK
53754 Sankt Augustin, Germany
E-mail: Spenke@gmd.de

ABSTRACT
FOCUS, the **F**eature-**O**riented **C**atalog **US**er interface, is an interactive table viewer for a common kind of table, namely the object-attribute table, also called cases-by-attribute table or relational table. Typical examples of these tables are the Roll Calls in BYTE where the features and test results of a family of hardware or software products are compared. FOCUS supports data exploration by a combination of a focus+context or fisheye technique, a hierarchical outliner for large attribute sets, and a general and easy-to-use dynamic query mechanism where the user simply clicks on desired values found in the table.

A PC/Windows implementation of FOCUS is publicly available (http://www.gmd.de/fit/projects/focus.html). It is suited for tables with up to a few hundred rows and columns, which are typically stored and maintained by spreadsheet applications. Since we use a simple data format, existing tables can be easily inspected with FOCUS.

With the rapidly increasing public interest in on-line services like the World Wide Web we expect a growing demand for access to on-line catalogues and databases. FOCUS satisfies this demand, allowing formulation of simple database queries with an interface as easy to use as a Web browser.

KEYWORDS: Dynamic Queries, Tables, Spreadsheets, Focus+Context Technique, Interactive Data Exploration.

INTRODUCTION
Frequently, product information can be presented as an *object-attribute table*. Electronic product catalogs, hotel guides, and even attribute-based selections of image material naturally fit into this model. With the growing interest in the WWW, the demand for adequate presentation of such material is increasing rapidly. Currently, there is no general, powerful, and easy-to-use mechanism available to visualize and explore such tables and to select and compare products.

Presenting the data as a *static table* becomes difficult if there are more than a dozen objects. Scrolling the data is cumbersome; objects can only be compared if near to each other; and it is difficult to obtain an overview about what is available.

Database queries can be used if the table is large, but queries require some knowledge about what can be expected, and in order to obtain good results the user must have a clear understanding as to which kinds of features are important. It is difficult to get an impression about what is not matched by a query, i.e., what could be found if the query were posed in another way.

We have developed FOCUS (the **F**eature-**O**riented **C**atalog **US**er interface). FOCUS enables the user to gain a flexible overview of an object-attribute table through a combination of a focus+context or fisheye technique and a hierarchical outliner for large attribute sets. With a few mouse clicks into the table, the user can formulate incremental database queries with immediate feedback after each step and thus restrict the table to the relevant subset of data. Although there are only very few interaction techniques, it is possible to express quite complex database queries.

FOCUS has been implemented in Visual C++ and runs on PC/Windows platforms. It is freely available and can be used as a helper application in a Web browser.

RELATED WORK
The Table Lens interface of Rao and Card [8] uses a focus+context technique to display large tables in a compressed form, but does not support queries. Our table interface uses a similar technique for the objects (columns): The width of a column is increased when selected (focused). Multiple selected columns correspond to multiple focal areas. However, in order to focus on the relevant attributes (rows), we use a hierarchical outliner. FOCUS combines the focus+context technique with incremental database queries and—as an extension—recognizes identical values in neighboring cells and prints them only once for a whole

Appeared in *UIST '96*
Proceedings of the ACM Symposium on
User Interface Software and Technology
Seattle, November 6-8, 1996
Copyright ACM 1996

range of columns. This measure considerably increases the readability of the table.

FOCUS follows principles formulated by Ahlberg and Shneiderman for Dynamic Query Filters [1] such as selection by pointing (not typing), immediate display of query results, tight-coupling, output-is-input, and progressive refinement of search parameters. However, FOCUS can handle grouping and arbitrarily nested AND/OR queries, and the table format supports the comparison of alternatives. Moreover, FOCUS is a generic, domain-independent tool.

Recently, Dynamic Queries has been generalized to be application independent, but the restrictions for the possible queries remain (Ahlberg and Wistrand [2], Chwelos and Mantei [3]). Fishkin and Stone [4] extend Dynamic Queries by Magic Lens filters in order to increase their expressiveness.

Goldstein and Roth [6] combine dynamic queries with data aggregation in order to enable more complex queries. However, the formulation of complex queries is more difficult than in FOCUS, because the Aggregate Manipulator and Dynamic Query components are not seamlessly integrated. Instead they are presented to the user as two different workspaces and explicit commands are necessary to transfer information between these workspaces. We will discuss later how the example queries of Goldstein and Roth can be expressed more easily with FOCUS.

Spreadsheets are classical tools to manipulate and compare tabular data. The filter mechanism of Excel [5] is similar to FOCUS, but queries in Excel are less expressive and there is no mechanism to delete irrelevant attributes from the table. Excel violates the principle of tight-coupling because the value menus for each attribute also display values that have already been filtered out.

The PERPLEX spreadsheet of Spenke and Beilken [9] uses logic programming and can simulate dynamic database queries but has been designed as a tool for end-user programming and is therefore too complicated for simple product-selection tasks.

THE INTERACTIVE TABLE

In order to illustrate the FOCUS user interface we have chosen a comparison of 92 printers as an example. The table appeared in the November 1994 issue of BYTE [7] and compares 51 different attributes (features), including vendor, price, resolution, supported interfaces, paper sizes, emulations, and benchmark results. Other printer tests show that there are many further attributes of potential interest, but, obviously, they would have consumed too much space. The original table printed in BYTE occupies four pages using a very small font.

Table Inspection

Figure 1 shows an overview of the printer table as displayed by FOCUS. Each column represents one of the printers and each row contains an attribute. A row can span several lines if the attribute has multiple values (e.g. *Supported Interfaces*). The table is shown in *overview mode* where FOCUS automatically adjusts the column width so that all printers are visible in the given window.

One of the column headers has been selected by the user as the *focal* column for closer inspection. Although many of the values in other columns are not readable, the user can at least estimate the number of printers for each of the five technologies. One can also observe that all laser and dot matrix printers are black and white, all thermal printers are color, and ink jets can be either color or black and white.

Displaying and Inspecting Compact Tables

Within the *Price* row (as in the other numerical rows) values are graphically indicated by the height of a bar; the corresponding numbers are only displayed if there is sufficient space available. This allows the user to make global observations even in highly compressed tables. For example, one can see that *ink jet* is the cheapest technology.

The column headers are displayed in vertical text if the column width is smaller than the height of the column headers, which can be adjusted by the user. Simple yes/no attributes are displayed as filled and empty circles to save space and to improve legibility.

In overview mode, adjacent and identical entries are combined into a single value spanning several columns, both to increase readability and to highlight similarities.

Multiple columns can be selected and unselected by clicking in the column header using shift and control as in a standard selection box. Similar to the focus+context technique described by Rao and Card [8], multiple selections result in multiple focal areas. Two or more distant table entries can thus be compared.

Dragging the mouse through the column header continuously changes the selection to the column under the mouse cursor. This is as easy as browsing through a scrollable selection list.

The attributes are structured hierarchically, like the lines of text in an outliner. Subtrees can be opened and closed by clicking on the triangle-shaped handles. In this way the user can focus on the relevant attributes. For example, in Figure 1 the attributes under *Emulations* are collapsed while those under *Interfaces* are expanded.

Sorting

In Figure 1 the printers are alphabetically sorted by *Technology*, indicated by the black arrow in the row header. As a consequence equal attribute values are joined and have to be displayed only once. By dragging the arrow (or by clicking on its final position) the table will be sorted by another attribute. Because we are using a stable sorting algorithm, fragmentation can be minimized throughout a series of sorts by different attributes. In our example, the table was sorted first by *Price*, second by *Color* and finally by *Technology*. Therefore the laser printers are still sorted by price.

If sorts are performed after a column has been selected, the user can watch where the selected printer is placed according to different criteria.

Figure 1: An overview of the printer table.

Normal Mode

The user can switch to *normal mode* using the *Overview* check box. In normal mode a larger (user-defined) column width is used and the table can be horizontally scrolled to view all entries. The table will automatically scroll to the first selected column, so that the focus is not lost. The normal column width can be adjusted by dragging the border between two columns.

Columns can also be selected in normal mode. They remain selected when the user switches back to overview mode, where they are shown highlighted and expanded (Figure 1).

TABLE RESTRICTION

The primary way to explore a FOCUS table is to restrict the number of products (columns) to an interesting subset.

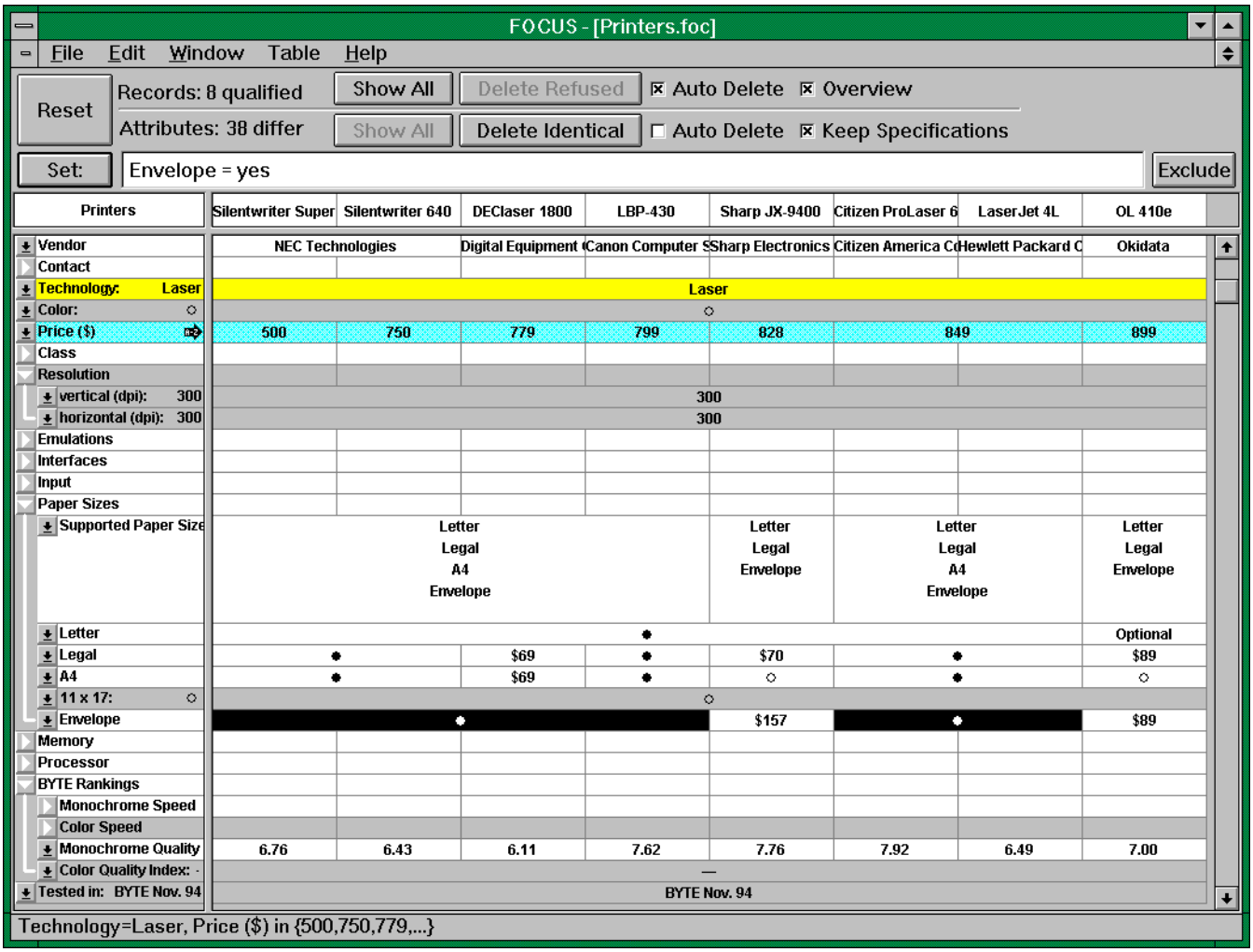
Constant Attributes

In Figure 1 the user has selected the value *Centronics* in the row *Supported Interfaces* (with a single mouse click). All cells in this row containing *Centronics* are highlighted. The user can see at a glance that almost all printers have a Centronics interface.

A value selection also proposes a corresponding query of the form <*attribute*> = <*value*> in the prompt line above the table. For example, in Figure 1 the query *Supported Interfaces* = *Centronics* is proposed. Pressing the *Set* button executes the query, i.e., removes all printers from the table that do not have a Centronics interface. The complete row appears with a yellow background to indicate an active restriction (e.g., the Technology row in Figure 2).

As an abbreviation, instead of first selecting a value and then pressing the *Set* button (or return key), the user can simply double click a value in the table. The most common way of interacting with FOCUS is by double clicking on desired values until only a few products of interest remain.

After executing such a query, all but the qualified records disappear from the table. In order to free a constant attribute again and to get back the deleted records, the user double clicks on the single yellow value now shown for the whole row. The user can also select the yellow value, observe that <attribute> = <All> is displayed in the prompt line, and press the *Set* button.



Printers	Silentwriter Super	Silentwriter 640	DEClaser 1800	LBP-430	Sharp JX-9400	Citizen ProLaser 6	LaserJet 4L	OL 410e
Vendor	NEC Technologies		Digital Equipment C	Canon Computer S	Sharp Electronics	Citizen America Co	Hewlett Packard C	Okidata
Contact								
Technology: Laser	Laser							
Color: ○	○							
Price ($)	500	750	779	799	828	849		899
Class								
Resolution								
vertical (dpi): 300	300							
horizontal (dpi): 300	300							
Emulations								
Interfaces								
Input								
Paper Sizes								
Supported Paper Size	Letter Legal A4 Envelope				Letter Legal Envelope	Letter Legal A4 Envelope		Letter Legal Envelope
Letter				●				Optional
Legal	●		$69	●	$70	●		$89
A4	●		$69	●	○	●		○
11 x 17: ○	○							
Envelope	●				$157	●		$89
Memory								
Processor								
BYTE Rankings								
Monochrome Speed								
Color Speed								
Monochrome Quality	6.76	6.43	6.11	7.62	7.76	7.92	6.49	7.00
Color Quality Index: -	—							
Tested in: BYTE Nov. 94	BYTE Nov. 94							

Figure 2: A restricted table (row *Technology* originally appears in yellow, row *Price ($)* in blue).

Subsets of Attribute Values

One can also select multiple attribute values within one row (using shift and/or control clicks as in a standard selection box). The proposed query then reads *<attribute> in {<value1>,<value2>, ...}*, i.e., pressing the *Set* button will restrict the table to records where the value of *<attribute>* is either *<value1>* or *<value2>*... Again a double click can be used as an abbreviation. The row will be displayed in blue to indicate that a subset has been chosen.

In Figure 2 only laser printers under $1000 are displayed. To achieve this, the value *Laser* was fixed by a double click. It is displayed on a yellow background. Next, the range of values smaller than 1000 was selected in the *Price* row and the *Set* button was pressed. The prices are displayed on a blue background.

The Attribute Menu

In order to release a subset restriction on an attribute again, the user has to select *<All>* from the attributes popup menu (Figure 3a). The menu appears when the mouse is pressed over the attribute name. Selecting *<All>* is also equivalent to freeing a constant attribute by double clicking in the yellow area where the constant is displayed.

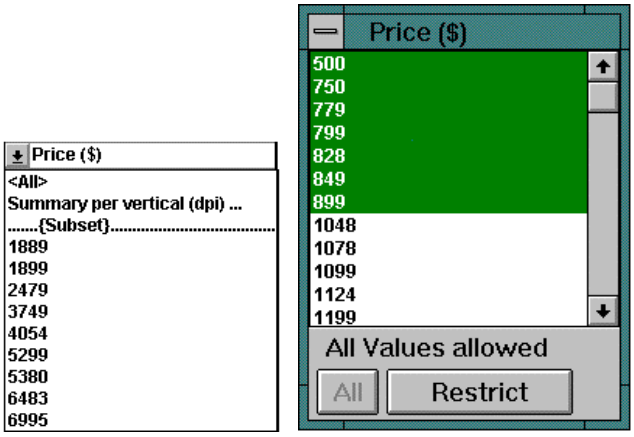


Figure 3a: An attribute menu. Figure 3b: A tear-off menu.

Attributes can also be restricted to a constant value by selecting that value from the attribute's popup menu. In this menu, only attribute values appearing in the current restriction are shown. For example, the result of restricting the selection to ink jets is that only the smaller prices are shown in the menu for the *Price* attribute. (We are following the principle of *tight coupling* formulated by Ahlberg and Shneiderman [1]). The values are always sorted. Therefore, in large tables it is sometimes easier to find a value in the menu than in the table.

Subset selection and restriction can also be performed in a tear-off menu (Figure 3b) which pops up when *...{Subset}...* is selected in an attributes menu. This modeless dialog box can also be used to permanently display the set of possible values of an attribute.

Excluding Values

Using the *Exclude* button on the right-hand side of the table, the user can exclude one or more selected values from the table. *Exclude* is equivalent to a negated *Set,* i.e., a restriction to the complementary set. Excluding a few values is sometimes easier than allowing all other values.

Query Representation

Note that FOCUS not only displays the result of a query but also the user specifications which led to the result: Lines are displayed in yellow or blue to indicate constant or restricted attributes. The selected subset can be viewed in the dialog box of an attribute. Furthermore, a summary of all user specifications is displayed in the status line at the bottom of the FOCUS window (Figure 2).

COMPLEX QUERIES

With a minimum of additional user-interface concepts, quite complex database queries can be defined and executed with a few mouse clicks. Even though they are simple to state, complex queries rarely have to be used because their results can often be directly observed in the table. This distinguishes FOCUS from database queries because all current objects are visible before a query is executed.

Disjunctions

When multiple values are selected in different rows, FOCUS proposes a disjunctive query of the form

<attribute1>=<value1> OR <attribute2>=<value2> ...

In Figure 4 the table has been restricted to thermal printers and the value 600 has been selected in both the *Resolution vertical* and *Resolution horizontal* rows. Pressing the *Set* button removes all records which do not have a resolution of 600 dpi in at least one of the dimensions.

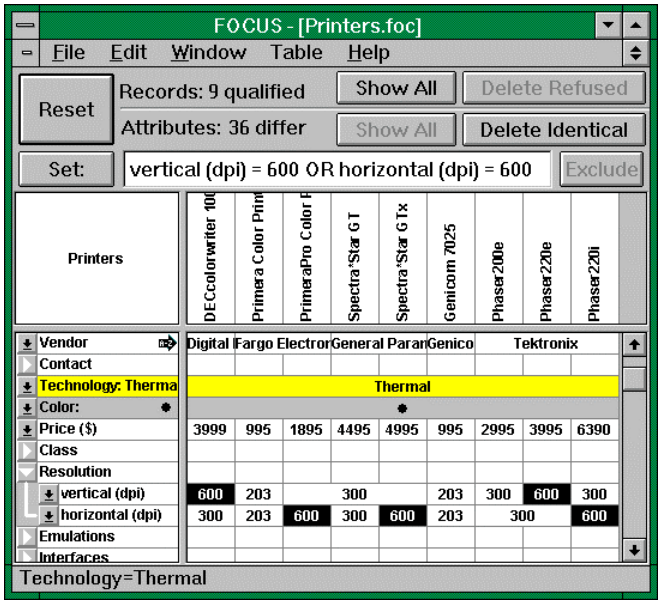


Figure 4: A disjunction.

Conjunctions

Because subsequent restrictions specified by the user are implicitly linked by the AND-Operator, the general form of a FOCUS query is a conjunction of disjunctions:

(<attribute$_A$> in {<value$_{A1}$>,<value$_{A2}$>, ...}
OR <attribute$_B$> in {<value$_{B1}$>,<value$_{B2}$>, ...}
OR ...)

AND

(<attribute$_C$> in {<value$_{C1}$>,<value$_{C2}$>, ...}
OR <attribute$_D$> in {<value$_{D1}$>,<value$_{D2}$>, ...}
OR ...)

AND ...

User-Defined Attributes

Whenever the user has restricted the table to an interesting subset of records, he can introduce a new user-defined attribute that is true for exactly the records in this subset. For example, the user can choose all laser printers with a high Postscript speed, select *Define New Attribute* from the *Table* menu and type *Fast Laser* into a dialog box as the name for the new attribute. The *Fast Laser* attribute will appear at the end of the table (see Figure 5). This is a convenient way to save the result of a query.

Grouping

When a table has been sorted according to an attribute, identical values of that attribute will be jointly displayed in one larger field spanning several columns, i.e., the records are *grouped* into classes with identical values. See for example *vertical (dpi)* in Figure 5.

Once a grouping of records into classes has been established (simply by moving the sort arrow), the user can then summarize the values of a different attribute for each class. For example, the user can ask for the mean price for each resolution class. The new attribute *Mean Price per vertical (dpi)* is defined by selecting the entry *Summary per vertical (dpi)* from the popup menu of *Price ($)* (see Figure 3a). A dialog appears where the user can choose between

- Minimum
- Maximum
- Sum
- Mean
- Count

The example in Figure 5 shows the surprising result that the mean price of fast laser printers in the 300 dpi class is much higher than in the 600 dpi class.

Queries with grouping are quite difficult to formulate in normal database query languages. In FOCUS a few mouse clicks are sufficient and grouping is visualized in a very natural way.

Of course, in order to determine the minimum or maximum of a group, it is not even necessary to define a new attribute: Sorting first according to *Price* and then according to *vertical (dpi)* will show the extreme prices at the edges of each group (as in Figure 5).

Nested Queries

Using user-defined attributes it is also possible to formulate more deeply nested queries or disjunctions of conjunctions such as *"Find all color printers under $2000 and all laser printers under $1000"*: The user can define two attributes *Cheap+Color* and *Cheap+Laser* and combine them with *OR* as described above.

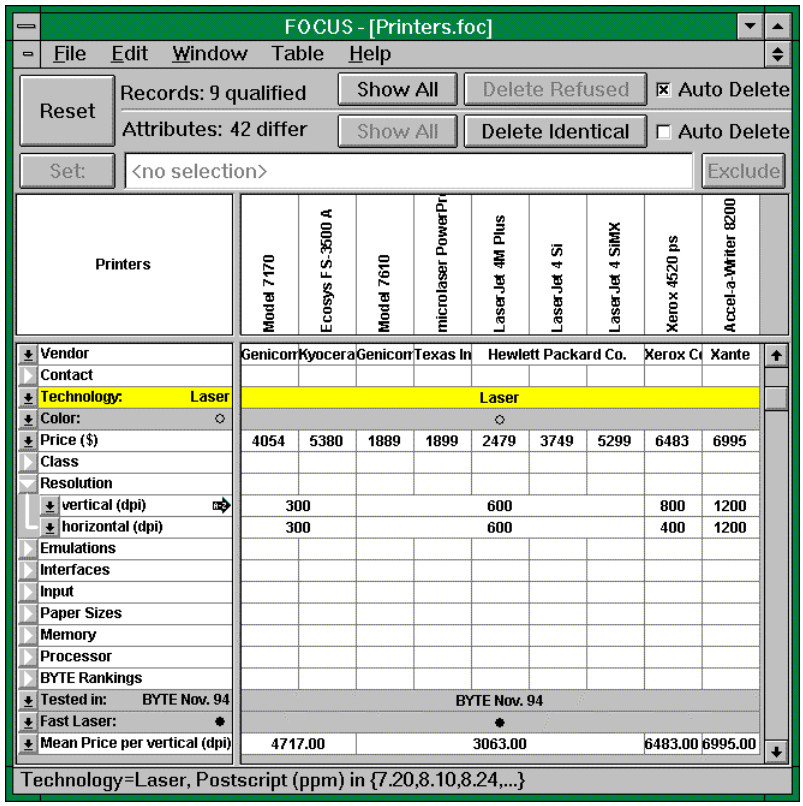


Figure 5: Two new user-defined attributes.

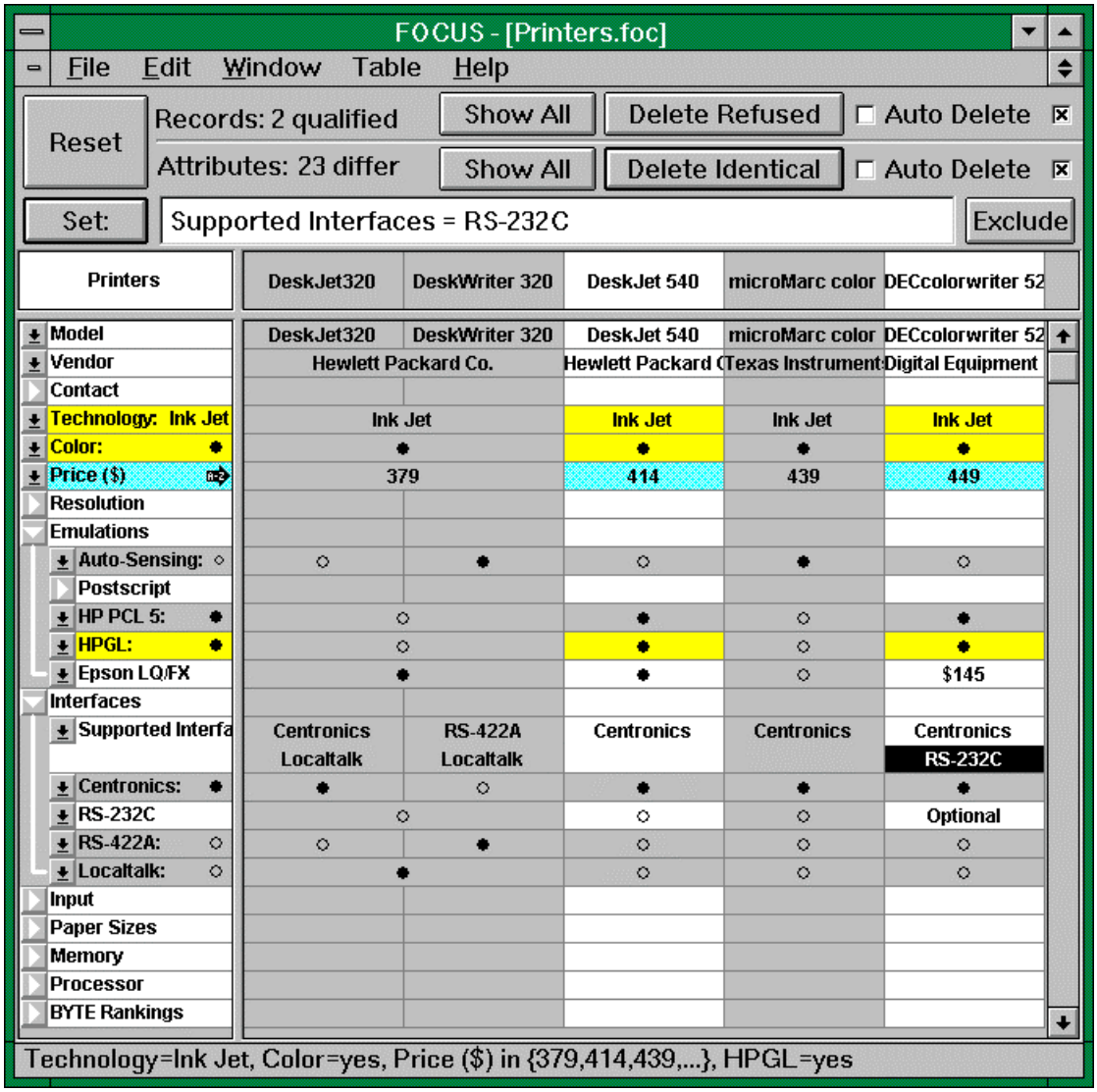


Figure 6: Displaying eliminated records and attributes.

AUTOMATIC VERSUS MANUAL DELETION

In the examples discussed so far the *Auto Delete* button for records was switched on so that records not matching are immediately removed from the table. Sometimes, however, it may be interesting for the user to see which of the records are filtered out before they are deleted. If *Auto Delete* mode is turned off, these refused records are just grayed out (see Figure 6). Later they can be explicitly deleted by pressing the *Delete Refused* button.

Figure 6 also contains some rows that have a gray background. FOCUS automatically determines which attributes still have different values in the restricted table. Attribute rows where all qualified records have the same value are grayed out. Also, section headers are grayed if the whole section is gray, whether they are open or closed, (see *Resolution* and *Color Speed* in Figure 2 for an example). This helps the user focus on the relevant differences between the remaining alternatives.

Pressing *Delete Identical* removes the gray rows. If *Auto Delete* is turned on for the attributes, this is performed automatically after each query.

As a default setting, we recommend *Auto Delete Records*, but not *Auto Delete Attributes*, because users can adjust to the fact that some records are removed after a query, but may be slightly confused when attributes suddenly disappear or move to a new position.

A common technique is to focus on a subset of records first and then to delete the irrelevant attributes and records. Continuing from that point with both *Auto Delete* buttons turned off, the window contents will remain stable: When queries are executed to further analyze the subset, some rows and columns may be grayed, but each cell and each value remains at its position.

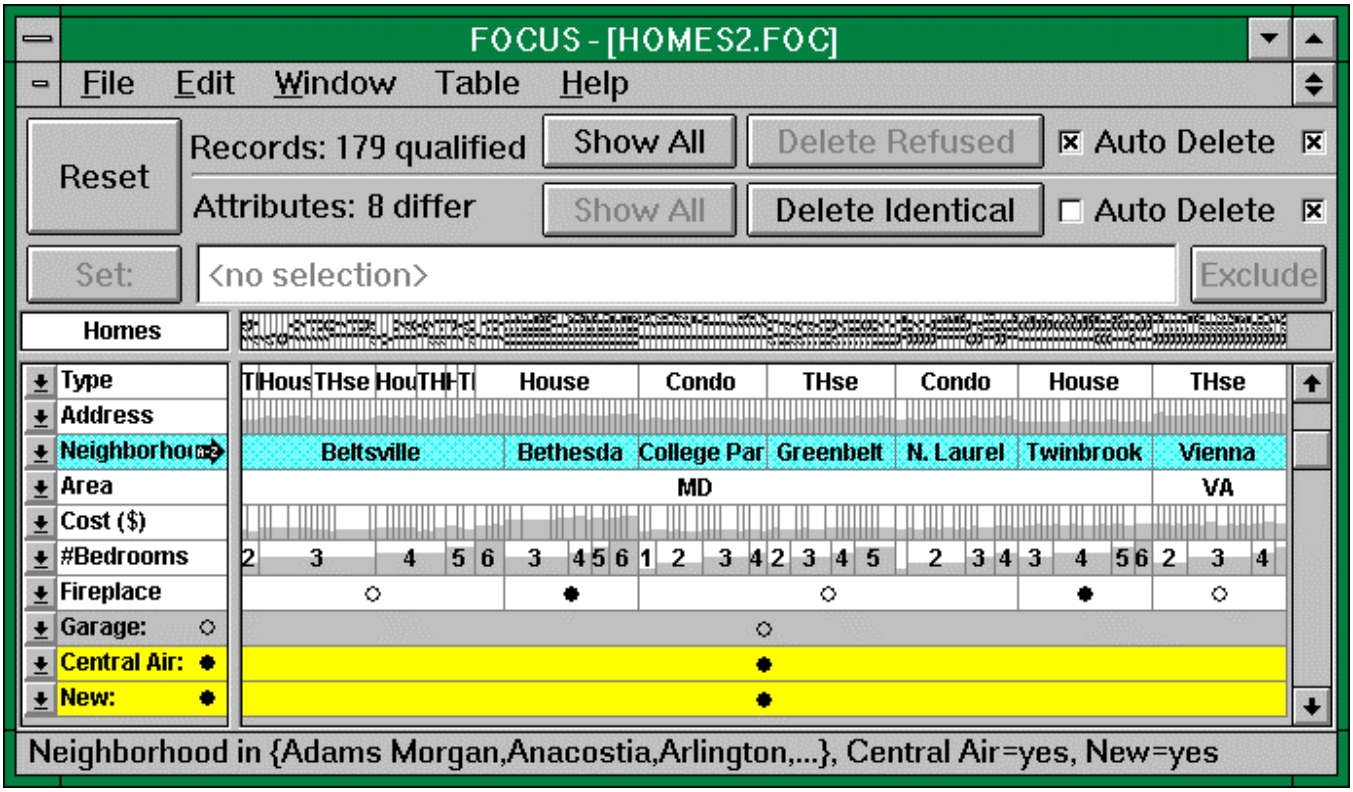


Figure 7: The Dynamic HomeFinder database.

DISCUSSION

To compare our approach to Dynamic Queries, we have also used FOCUS to explore the database of the Dynamic HomeFinder demo [1]. Figure 7 shows a part of the database.

While the Dynamic HomeFinder can display the complete data of only one record at a time, FOCUS gives a very informative overview of the data. Without performing any further queries one can discover the following facts about the shown homes:

- They are distributed over 7 neighborhoods.
- All neighborhoods but one are in the MD area.
- In Bethesda there are only houses.
- In College Park there are only condos.
- Bethesda is the most expensive neighborhood.
- Fireplaces are found in only two neighborhoods.
- Most homes in Beltsville have 3 Bedrooms.
- No house has more than 6 bedrooms.

And many more ...

Of course, it is appropriate to show the geographical locations of houses for example with a starfield display, but a general tool such as FOCUS cannot create this kind of application dependent visualization.

Goldstein and Roth [6] use a similar real estate database to discuss two scenarios.

First Scenario

In the first scenario Jennifer searches for houses between $100,000 and $150,000 with a lot size larger than 5000 sq. ft. in the neighborhoods Shadyside or Squirrel Hill *or* (in the same price range) a lot size larger than 8000 sq. ft. in Point Breeze. This disjunction of two queries is performed by first defining two new aggregates, *SqHill-Shady* and *PtBreezeBigLot,* for the two queries. Then a third new aggregate, *Interest Houses*, is introduced to construct the union of the two aggregates.

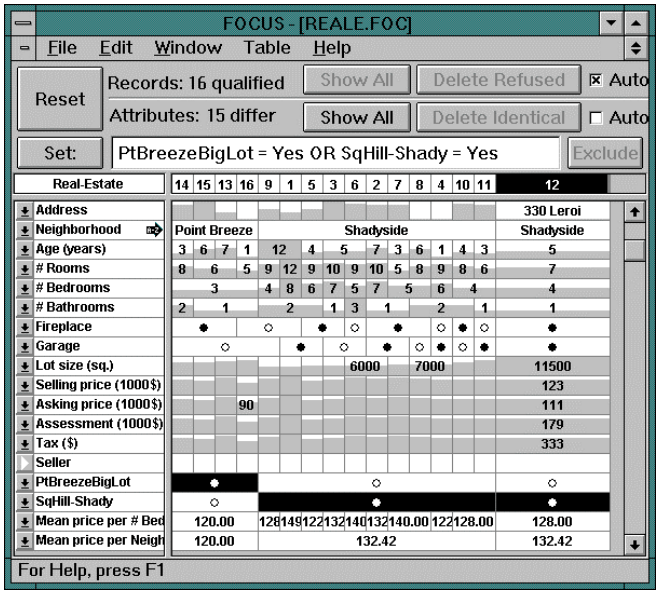


Figure 8: First scenario of Goldstein and Roth [6].

Similarly with FOCUS, one can define two new user-defined attributes: The first query is performed and the result is stored as a new attribute. Next, the second query is formulated as a slight modification of the first, and a second attribute is defined. There is no need to define a third attribute, because an OR-query can be easily performed by selecting *yes* (filled circle) in the two rows for the new attributes. With FOCUS, fewer user actions are required because there is no need for commands that transfer information between the Aggregate Manipulator and the Dynamic Query component.

Using FOCUS Jennifer can actually solve her problem much more easily without defining new attributes and performing a formal OR-query (see Figure 8). She could simply restrict the table to the desired price range and the three neighborhoods. If the table were then sorted by *Lot Size* and afterwards by *Neighborhood*, Jennifer would see the desired houses at a glance. This technique, which is closer to browsing than to formal database queries, is more convenient for the end user and is ideally supported by FOCUS.

Also in the first scenario, summary statistics are generated for the remaining houses. Figure 8 shows how the complete information, distributed over several display areas in the system of Goldstein and Roth [6], is displayed in one clearly arranged table. The user only had to sort by *#Bedrooms* and then to select *Summary per #Bedrooms* from the menu of *Cost ($)*. In Figure 8 the mean price per neighborhood has also been determined.

Second Scenario

In the second scenario John asks for the company and then the sales agent who sold the most houses from a given subset. To answer this question, new aggregates are defined. In FOCUS however, two mouse clicks are sufficient to sort first by *Sales Agent* and then by *Company*: John can see at a glance which agent of which company spans the most columns.

A WORLD WIDE WEB TABLE BROWSER

FOCUS is ideally suited as a browser for product information to be distributed on the World Wide Web. Tables on the WWW share the same problem with their counterparts on paper: if they grow too large, they become very unwieldy to manage.

Currently, the only way to display databases on the WWW is to use a text field to enter a query and to display the query results as a new page. With FOCUS, medium-sized databases can be included as full tables within a page and can be interactively viewed in place.

For this purpose, FOCUS tables have to be registered as a MIME type. The tables are quite small compared to typical graphical images. For example, the printer table presented here occupies only 32 kB as an ASCII file.

There are three different ways of viewing tables on the WWW with FOCUS:

- Configuring the WWW browser to use FOCUS as an external viewer for tables. This is the way FOCUS is currently implemented. The advantage is that FOCUS is just an arbitrary application and is only loosely related to the respective Web browser. The disadvantage is that the table resides in a different window and provides no link back into the Web.

- Implementing FOCUS as a plug-in for the Web browser. In this solution the table viewer is more directly integrated into the Web browser, but the disadvantage is that the integration is browser-dependent.

- Implementing FOCUS as a JAVA applet. This solution makes FOCUS platform-independent and usable with most future Web browsers.

We would like to create a new standard for handling product selection in the WWW. It is therefore in our interest to make FOCUS as widely available as possible.

REQUIREMENTS FOR DATA STRUCTURE

FOCUS is applicable to a wide range of tables. The basic precondition is that the table has to be indexed by products and attributes. Such a table is similar to a flat relation.

If there are multiple relations concerning the same entities, these relations have to be joined to make the result viewable with FOCUS. Therefore, most relational databases will be good candidates for presentation by FOCUS.

For product descriptions, subclass relationships frequently exist between different product categories. That means that for a specific subclass of products, further descriptive attributes are needed. In FOCUS, several such subclasses can be integrated into a single table for comparison by taking the union of all attributes and by leaving the attribute values empty for all undefined attributes. In the table they will be displayed as "—" (not applicable). Usually the hierarchical attribute system can be used in such cases to improve clarity. Attributes for a subclass are grouped under a particular header. The user can hide all these attributes with a single click and redisplay them if objects of this subclass are selected.

The input to FOCUS is a simple tab-delimited ASCII file. Such a file can be generated with a minimum of effort from many popular applications, such as spreadsheets, simple databases, card files, and even tables in text files.

APPLICATION FIELDS

There are a number of applications that illustrate how FOCUS can be used. The most common examples are technical product databases. In these situations, FOCUS supports the complex selection of suitable products that are described by a multitude of attributes with no clearly defined evaluation procedure.

In addition to printers and other computer equipment, typical product categories include cars, stereo equipment, washing machines, etc.

Mail-order catalogs include many different products with some common attributes such as price, order number, and delivery time, but also contains products with some product-specific attributes that only apply to certain product classes, such as color, size, and sex for articles of clothing. FOCUS is well suited for this kind of large sparse tables, because attributes can be grouped in hierarchical subsections and irrelevant attributes disappear automatically from the table after selecting a special product class.

A similar situation exists for a directory of camping sites or a hotel database. Again, this kind of data can be described by a number of different attributes, which have vastly different importance for different customers.

Pharmaceutical products databases are also a domain where the comparison of different products is very important. Often there is no optimal drug, so it is necessary to compare different side effects.

CONCLUSION

We have developed a simple, powerful, and efficient interactive viewer for tabular data which is ideally suited for data exploration and interactive query formulation. FOCUS seamlessly combines ideas from the Table Lens [8], Dynamic Queries [1], and the Aggregate Manipulator [6] into one consistent concept.

The principal features of FOCUS are:

- The whole table (or any subset) is completely visible on a single page in a highly compressed but still readable form.
- The table can be restricted to an interesting subset through dynamic queries specified within the table using simple mouse clicks.
- Complex queries can also easily be specified but are rarely necessary because of the flexible overview, the sorting capability, and the hierarchical attribute list.

FOCUS is freely available in the World Wide Web (http://www.gmd.de/fit/projects/focus.html) and can be used as a helper application of a WWW browser.

REFERENCES

1. Ahlberg, C. and Shneiderman, B., Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, Apr 24–28, 1994), pp. 313–317.
2. Ahlberg, C. and Wistrand, E., IVEE: An Environment for Automatic Creation of Dynamic Queries Applications. In *Conference Companion of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Denver, CO, May 7-11, 1995), pp. 15-16.
3. Chwelos, G. and Mantei, M., Design Space of a Generic Interface for Filtering and Displaying Database Query Results. In *Adjunct Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, Apr 24–28, 1994), pp. 175–176.
4. Fishkin, K. and Stone, M.: Enhanced Dynamic Queries via Movable Filters. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Denver, CO, May 7-11, 1995), pp. 415-420.
5. Excel User Manual, Version 5.0, Microsoft, 1993.
6. Goldstein, J. and Roth, S. F., Using Aggregation and Dynamic Queries for Exploring Large Data Sets. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, Apr 24–28, 1994), pp. 23–29.
7. Kane, J. and McDonough, J., 92 Printers go to Battle. In *BYTE*, November 1994, pp. 262-283.
8. Rao, R. and Card, S. K., The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, Apr 24–28, 1994), pp. 318–322.
9. Spenke, M. and Beilken, C., PERPLEX: A Spreadsheet Interface for Logic Programming. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Austin, TX, Apr 30–May 4, 1989), pp. 75–80.